F5
Secure Sockets

EDA095 Nätverksprogrammering

Roger Henriksson
Datavetenskap
Lunds universitet

Java Secure Sockets Extension – JSSE

Secure Sockets Layer – SSL
Transport Layer Security - TLS

Protokoll och algoritmer för säker kommunikation via TCP. TLS nyare vidareutveckling av SSL.

Tillhandahåller krypterade förbindelser samt autentisering.

JSSE tillhandahåller klasser som gör det (relativt) enkelt att upprätta säkra förbindelser mellan två javaprogram eller ett javaprogram och en webbserver eller en webbläsare.

Säker kommunikation – krav

- Krypterad överföring – vi vill inte att någon ska kunna avlyssna vad vi sänder.
- Autentisering – vi vill vara säkra på att den vi kommunicerar med faktiskt är den vi tror.

Kryptering

Delad-nyckel-kryptering

- Båda sidor har i förväg enats om en kryptonyckel som används för att kryptera data som överförs.
- Enkelt och effektivt!
- Besvärligt att dela nyckeln på ett säkert sätt.

Öppen-nyckel-kryptering

- En publik och en privat nyckel. Mottagarens publika nyckel används för kryptering och mottagaren avkodar det med sin privata nyckel.
- Inga i förväg delade nycklar.
- Beräkningsintensiva algoritmer.

Hybridlösning för kryptering

- Öppen-nyckel-kryptering används för överföring av en delad nyckel.
- Den delade nyckeln används för kryptering av den data som ska överföras.

Fördelar

- Ingen i förväg delad nyckel.
- Snabb och effektiv kryptering av data.

Autentisering

Hur vet vi att den vi kommunicerar med verkligen är den vi tror? Undvik mannen-i-mitten-attacker!

1. Avsändaren krypterar ett meddelande med sin *privata* nyckel.
2. Meddelandet sänds till mottagaren.
3. Mottagaren avkodar meddelandet med mottagarens *publika* nyckel. Om klartext erhålles var det ägaren till den publika nyckeln som sände meddelandet.

Hur vet vi att ingen har bytt ut nyckeln på vägen?

Certifikat

- Publika nycklar lagras hos en betrodd tredje part – en certifikatsutgivare.
- Mottagaren jämför den mottagna publika nyckeln med nyckeln hos den betrodda tredje parten.
- Mannen-i-mitten-attack fortfarande teoretisk möjlig, men mycket svårare.

Paket

- `javax.net.ssl` – abstrakt API
- `javax.net` – abstrakt API
- `java.security.cert` – certifikat
- `com.sun.net.ssl` – faktisk implementation

Nya klasser för att skapa och konfigurera en krypterad förbindelse. Därefter används de vanliga operationerna för sockets och strömmar.

SSLSocketFactory

Skapar en krypterad socket (**`SSLSocket`**).

SSLSocketFactory

```
public static getDefault();

public String[] getSupportedCipherSuits();
public String[] getDefaultCipherSuits();

public Socket createSocket(String host, int port)
             throws IOException, UnknownHostException;
public Socket createSocket(InetAddress host, int port)
             throws IOException, UnknownHostException;
```

Krypteringsalternativ

Olika implementationer stödjer olika kombinationer av kryptoalgoritmer och autentiseringsmetoder.

DEMO – CipherSuites.java

Alla alternativ som stöds är inte aktiverade från början. Vi kan välja vilka alternativ vi vill tillåta.

`TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`

Protokoll, nyckelutbytesmetod, krypterings-algoritm och checksumma. Anses vara säkrast.

Utvecklingsexempel

- Utveckla en mycket enkel webbserver som stödjer krypterade HTTP-uppkopplingar (https).
- Gör så att man kan nå servern via Firefox och https-protokollet.
- Skriv en enkel klient som kan ladda ner en webbsida från servern.

DEMO - WWW

Certifikat

Vi måste skapa en fil innehållande certifikat (autentiseringsinformation / kryptonycklar).

Informationen används för att upprätta en säker förbindelse då en klient ansluter.

I JSSE ingår ett verktyg för att skapa en sådan certifikatsfil - **`keytool`**.

DEMO - keytool

Se kurshemsidan för komplett kommando.

Klasser för att starta/konfigurera servern

KeyStore

```
public static KeyStore getInstance(String type)
                                  throws KeyStoreException;
public void load(InputStream stream,char[] password)
      throws IOException, NoSuchAlgorithmException,
                                        CertificateException;
```

KeyManagerFactory

```
public static KeyManagerFactory getInstance(String
            algorithm) throws NoSuchAlgorithmException;
public KeyManager[] getKeyManagers();
```

Klasser…, fortsättning

SSLContext

```
public static SSLContext getInstance(String protocol)
                    throws NoSuchAlgorithmException;
public void init(KeyManager[] km, TrustManager[] tm,
    SecureRandom random) throws KeyManagementException;
public SSLServerSocketFactory getServerSocketFactory()
                          throws IllegalStateException;
```

SSLServerSocketFactory

```
public SSLSocket createServerSocket(int port)
                                  throws IOException;
```

SSLServerSocket (extends ServerSocket)

SecureWWW

DEMO – SecureWWW med Firefox

Klientklasser

SSLSocketFactory

```
public static SSLSocketFactory getDefault();
public SSLSocket createSocket(String host,int port)
            throws IOException, UnknownHostException;
```

SSLSocket (extends Socket)

DEMO – SslSocketClient

Ange tillförlitliga certifikat

Certifikatet är självsignerat och accepteras inte av klienten.

Mha **`keytool`** skapar vi en publik version av serverns certifikat och för över det till klienten.

Se kurshemsidan för detaljer.

Vi startar klienten med:

```
java -Djavax.net.ssl.trustStore=trusted.jks SslSocketClient
```

DEMO – SslSocketClient med certifikat